Digital hyperplane fitting

Phuc Ngo

Digital hyperplane fitting

Phuc Ngo[1]

Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
hoai-diem-phuc.ngo@loria.fr

Abstract. This paper addresses the hyperplane fitting problem of discrete points in any dimension (*i.e.* in $\mathbb{Z}^d$). For that purpose, we consider a digital model of hyperplane, namely *digital hyperplane*, and present a combinatorial approach to find the optimal solution of the fitting problem. This method consists in computing all possible digital hyperplanes from a set $\mathbf{S}$ of n points, then an exhaustive search enables us to find the optimal hyperplane that *best fits* $\mathbf{S}$. The method has, however, a high complexity of $O(n^d)$, and thus can not be applied for big datasets. To overcome this limitation, we propose another method relying on the Delaunay triangulation of $\mathbf{S}$. By not generating and verifying all possible digital hyperplanes but only those from the elements of the triangulation, this leads to a lower complexity of $O(n^{\lceil \frac{d}{2} \rceil + 1})$. Experiments in 2D, 3D and 4D are shown to illustrate the efficiency of the proposed method.

Keywords: Optimal consensus · Exact computation · Discrete optimization · Optimal fitting · dD Delaunay triangulation.

1 Introduction

Data fitting is the process of matching a set of data points with a model, possibly subject to constraints. This is an essential task in many applications of computer vision and image analysis; *e.g.* shape approximation [23, 32], image registration [31, 33], image segmentation [19, 21]. In this context, the mostly considered models are the geometric ones such as a line, a circle in 2D or a plane, a surface in 3D. Among the models, the linear one has received greatest attention in theory and practice as many nonlinear models can be rearranged to a linear form by a suitable transformation of the model formulation [17, 30]. In this paper, we are interested in the fitting problem of hyperplane –a linear model– for a set of discrete points $\mathbf{S}$ in any dimension; *i.e.* $\mathbf{S} \subset \mathbb{Z}^d$ for d is the dimension of space. Then, this problem can be formulated as an optimization problem in which we find the parameters of the hyperplane that *best fits* $\mathbf{S}$, namely *optimal solution*. It is clear that this problem depends on how we define the hyperplane model and the criteria for the best fitting, namely cost function of the optimization process. In practice, hyperplane fitting has a great interest in applications of classification for object detection and recognition [9].

Several works have been proposed in this context. We can mention, for instance, the methods based on regression [4, 11, 29, 30]; *e.g.* least squares, weighted

least-squares, least-absolute-value regression and least median of squares (LMS). Generally, these approaches consider a hyperplane in the Euclidean space $\mathbb{R}^d$ and find the model that minimizes the sum of the geometric distance from all given points to the model. However, the provided solution is known to be unstable and sensitive to large deviant data points, namely *outliers* [24]. Other well-known approaches for fitting hyperplane using voting scheme include the Hough Transform (HT) [15, 18], the RANdom SAmple Consensus (RANSAC) [16] and associated variations [12]. More precisely, HT considers a dual space, namely *Hough parameter space* of the input set $\mathbf{S}$. This space is discretized into cells and used as an accumulator of votes from points of $\mathbf{S}$. Typically, for each point $\mathrm{x} \in \mathbf{S}$, a vote is added to cell in the accumulator that could generate from x. Then, the optimal solution is computed from the cell having the maximal of votes. RANSAC method chooses d points from $\mathbf{S}$ at random to form a candidate hyperplane passing through these points. Then, it calculates the distance of every points of $\mathbf{S}$ to the candidate hyperplane, and the points within a threshold distance are considered to be in the *consensus set* of the hyperplane. A score associated to the hyperplane is computed based on its consensus set; *e.g.* the size of the consensus set. This process is iterated a certain times and reports the hyperplane of maximal score as the optimal solution; *i.e.* the best-fitting hyperplane for $\mathbf{S}$. Both HT and RANSAC are simple, efficient and robustness to outliers. They, however, have a computational complexity growing with the number of model parameters [12, 15, 16, 18]. In addition, the above approaches were originally designed for points in $\mathbb{R}^d$ using the Euclidean hyperplane model. Of course, the discrete space of $\mathbb{Z}^d$ is a subspace of $\mathbb{R}^d$, then the fitting problem can be solved using these approaches. However, because of their continuous consideration, applying them for points of $\mathbb{Z}^d$ requires the use of floating point numbers which may induce the numerical error. Furthermore, due to the discrete nature of points in Z^d, computing an actually optimal solution of discrete points is practically impossible in continuous space as there is an infinity of solutions.

Under the assumption of input points in discrete space of $\mathbb{Z}^d$, this paper addresses the problem of hyperplane fitting in a fully discrete context. More specifically, we consider the digital model of hyperplane of $\mathbb{Z}^d$, namely *digital hyperplane*, and propose methods for digital hyperplane fitting using exact computation. For that purpose, we first present a combinatorial approach for solving this problem. The method consists in generating all the possible digital hyperplanes associated to a set of n point $\mathbf{S} \subset \mathbb{Z}^d$. Contrarily to $\mathbb{R}^d$, this set –despite a potentially high complexity– remains finite and thus allows an explicit exploration to find the optimal solution via a discrete optimization scheme. The method guarantees the global optimality, it has however a computational complexity of $O(n^d)$. In practice, it is unsolvable for $n = 10^6$ in 3D[1]. This high complexity practically forbids its use for big dataset.

In order to solve this fitting problem in a practical context, we propose a new method to find locally optimal solution. The method is based on a heuristic

[1] Supposing it needs $1\mu s$ for generating and testing one hyperplane, then it takes about $3 * 10^7$ years to find the optimal solution.

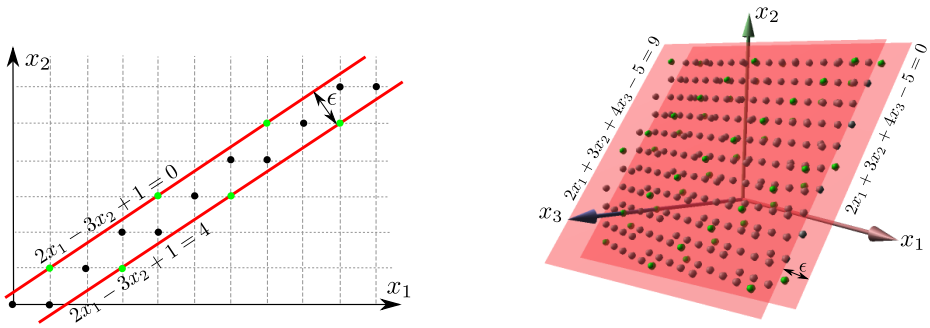


Fig. 1: Digital hyperplanes in 2D and 3D. Left: a digital line and right: a digital plane.

involving the Delaunay triangulation [13, 27]. Basically, it consists in computing the Delaunay triangulation of the input points, then using the triangulated elements to generate hyperplanes and find an optimal solution for the fitting problem stated above. It should be mentioned that Delaunay triangulations are used in numerous applications of computational geometry [8], geometric modelling [6] and computer graphics [3, 14]. It is not only well-known for its optimal properties [2, 22, 27] but also for its advantage to be incrementally computed in $O(n^{\lceil \frac{d}{2} \rceil + 1})$ complexity [7, 10].

2 Preliminaries

2.1 Digital hyperplane

An affine hyperplane in Euclidean space $\mathbb{R}^d$ of dimension $d \geq 2$ is defined by the set of points $\mathrm{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ satisfying the following equation:

$$\mathcal{H} = \{\mathrm{x} \in \mathbb{R}^d : \sum_{i=1}^{d} a_i x_i + a_{d+1} = 0\}, \tag{1}$$

with $a_i \in \mathbb{R}$ are coefficients of the hyperplane, w.l.o.g. $\mathcal{H}$ can be unambiguously represented by its parameters and denoted by $\mathcal{H} = (a_i)_{i=1}^{d+1}$. In other words, a hyperplane is the solution of a single linear equation (Eq. 1). Lines and planes are respectively hyperplanes in 2 and 3 dimensions.

The digitization of hyperplane in the discrete space of $\mathbb{Z}^d$ is called *digital hyperplane*, and defined as follows. Note that this definition is similar to the one in [28] with a slight difference at the double less than or equal to ($\leq$) in Eq. 2.

Definition 1. *A* digital hyperplane *in* $\mathbb{Z}^d$, $d \geq 2$, *is defined by the set of discrete points* $\mathrm{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{Z}^d$ *satisfying the inequalities:*

$$\mathcal{DH}_\omega = \{\mathrm{x} \in \mathbb{Z}^d : 0 \leq \sum_{i=1}^{d} a_i x_i + a_{d+1} \leq \omega\}, \tag{2}$$

with $a_i \in \mathbb{Z}$ *are coefficients of the digital hyperplane, and* $\omega \in \mathbb{Z}$ *a given constant.*

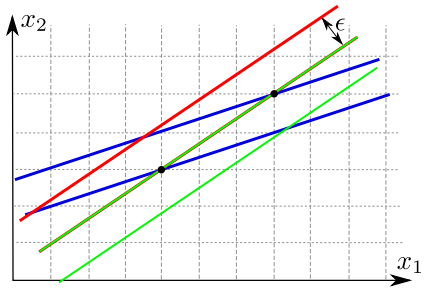


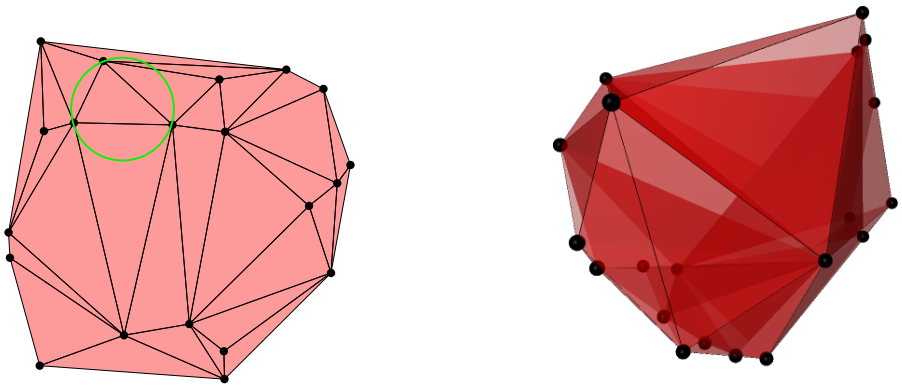
Fig. 2: Different digital lines of thickness ω passing through two points (in black).

Such a digital plane $\mathcal{DH}_\omega$ can be represented by its parameters and denoted by $\mathcal{DH}_\omega = (a_i)_{i=1}^{d+1}$. Geometrically, $\mathcal{DH}_\omega$ is a set of discrete points lying in between two parallel hyperplanes $\sum_{i=1}^{d} a_i x_i + a_{d+1} = 0$ and $\sum_{i=1}^{d} a_i x_i + a_{d+1} = \omega$; these two parallel hyperplanes are called the *support hyperplanes*, and the points that are on the support hyperplanes are called *support points* of $\mathcal{DH}_\omega$. The distance between the two hyperplanes is $\epsilon = \frac{w}{\sqrt{\sum_{i=1}^{d} a_i^2}}$ which refers to the *euclidean thickness* of $\mathcal{DH}_\omega$, while w refers to *arithmetical thickness* of $\mathcal{DH}_\omega$ [28]. An example in 2D and 3D is given in Fig. 1.

From linear algebra, a hyperplane in dimension d is a $(d-1)$-dimensional subspace; *i.e.* it is defined by $(d-1)$ linearly independent vectors. These $(d-1)$ vectors can be created from d distinct points of $\mathcal{H}$. In other words, given any d points in the hyperplane in general linear position, *i.e.* they are all $(d-1)$ linearly independent, there is a unique hyperplane of $\mathbb{R}^d$ passing through them. In case of $\mathbb{Z}^d$, from Eq. 2, we also need d linearly independent support points to determine a digital hyperplane. However, contrarily to the Euclidean space, for a given set of d support points and a value ω, the digital hyperplane $\mathcal{DH}_\omega$ passing though these points is not unique. This is illustrated in Fig. 2.

2.2 Delaunay triangulation

The *Delaunay triangulation* was introduced by Boris Delaunay in [13]. It is initially defined for a given set of n points $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{R}^2 \mid i = 1..n\}$ as a triangulation $\mathcal{DT}(\mathbf{S})$ such that the circumcircle associated to any triangle in $\mathcal{DT}(\mathbf{S})$ does not contain any other points of $\mathbf{S}$ in its interior. In other words, a Delaunay triangulation fulfills the *empty circle property* (also called *Delaunay property*): the circumscribing circle of any triangle of the triangulation encloses no other data point. Such a triangulation can be seen as a partition of the convex hull of $\mathbf{S}$ into triangles whose vertices are the points of $\mathbf{S}$, and it maximizes the minimum angle of all the angles of the triangles in $\mathcal{DT}(\mathbf{S})$. An illustration is given in Fig. 3(a). It should be mentioned that in some degenerate cases, the Delaunay triangulation is not guaranteed to exist or be unique; *e.g.* for a set of linear points there is no Delaunay triangulation, for four or more points on the same circle the Delaunay triangulation is not unique.



(a) (b)

Fig. 3: Illustration of Delaunay triangulation in 2D (left) and 3D (right).

By considering circumscribed spheres, the Delaunay triangulation extends to three and higher dimensions, namely *dD Delaunay triangulation* [27] for d is the dimension of space (see Fig. 3(b) for an example in 3D). Then, we call an *i-face* for $i \in [0, d]$ is an element of $\mathcal{DT}(\mathbf{S})$ containing $i{+}1$ vertices of $\mathbf{S}$. Then, a vertex is a 0-face, an edge is a 1-face, a triangle is 2-face, a tetrahedron is a 3-face, a ridge is a (d-2)-face, a facet is a (d-1)-face and a full cell is a d-face. In [22, 27], discussions on optimal properties of $\mathcal{DT}(\mathbf{S})$ in d dimension have been presented such as the maximum min-containment radius, uniformity of size and shape. In particular, as mentioned in [27] the dD Delaunay triangulation can be transformed into a convex-hull problem in dimension $d+1$. Henceforth, convex-hull algorithms can be used to obtain the Delaunay triangulation. In this context, there exist efficient and incremental algorithms [5, 10, 27] to construct the Delaunay triangulation. Furthermore, it is shown in [26] that the dD Delaunay triangulation of n points in $\mathbb{R}^d$ contains $O(n^{\lceil \frac{d}{2} \rceil})$ faces.

In this paper, we consider the dD Delaunay triangulation for the set of discrete points $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{Z}^d \mid i = 1..n\}$, and use the implementation of Delaunay triangulation proposed in CGAL [10] because of its robustness, efficiency, ease of use and flexibility. It is shown in [7] that the worst case complexity, without spatial sort of input points, of the method is $O(n^{\lceil \frac{d}{2} \rceil + 1})$. With spatial sort and random points, one can expect a much better complexity of $O(n \log n)$.

3 Digital hyperplane fitting

From Def. 1, we can describe our fitting problem for a set of discrete points $\mathbf{S}$ as finding a digital hyperplane $\mathcal{DH}_\omega = (a_i)_{i=1}^{d+1}$ of given ω that encloses the most number of points of $\mathbf{S}$. Such a hyperplane is called *optimal hyperplane*, the points of $\mathbf{S}$ belonging to $\mathcal{DH}_\omega$, *i.e.* $\mathrm{x}_i \in \mathbf{S} \cap \mathcal{DH}_\omega$, are called *inliers*, the other points of

$\mathbf{S}$ are called *outliers*. In other words, the digital hyperplane fitting aims to solve an optimization problem being expressed as maximizing the number of inliers.

Definition 2. *Given* $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{Z}^d \mid i = 1..n, \text{ for } n \geq d\}$ *and a constant* $\omega \in \mathbb{Z}$. *The best fitting hyperplane of* $\mathbf{S}$ *is defined as*

$$\mathcal{DH}^*_\omega = \underset{\mathcal{DH}_\omega \in \mathbb{F}(\mathbf{S})}{\arg\max} \ \{\mathbf{S} \cap \mathcal{DH}_\omega\}$$

where $\mathbb{F}(\mathbf{S})$ *is the search space and it contains the set of all hyperplanes of given* ω *generated from* $\mathbf{S}$.

Due to the discrete nature of the problem, it should be mentioned that the search space $\mathbb{F}(\mathbf{S})$ can be huge but finite as $\mathbf{S}$ is finite and the points $\mathrm{x}_i \in \mathbf{S}$ have finite coordinates. Roughly speaking, a brute-force search within $\mathbb{F}(\mathbf{S})$ would lead to a globally optimal solution for the digital hyperplane fitting of $\mathbf{S}$ in Def. 2. Finding $\mathbb{F}(\mathbf{S})$ in 2D (resp. 3D) case is solved in [34]. More specifically, using rotation and translation techniques, it is proved that a digital lines (resp. planes) can be determined with at least 2 (resp. 3) support points. Therefore, the whole search space $\mathbb{F}(\mathbf{S})$ can be constructed from all possible pairs (resp. triplets) of points in $\mathbf{S}$ for digital line (resp. plane) fitting.

Property 1 ([34]). Given a set of points $\mathbf{S} \subset \mathbb{Z}^2$ (resp. $\mathbb{Z}^3$), and a set of inliers, namely *consensus set*, for a given digital line (resp. plane). It is possible to find a new digital line (resp. plane) with the same consensus set, such that it has at least 2 (resp. 3) inliers as support points.

This is clearly understandable as a digital line (resp. plane) can be computed from two (resp. three) support points. This result can be extended to higher dimension thanks to the very definition of digital hyperplane (see Def. 1).

Property 2. Given a set of points $\mathbf{S} \subset \mathbb{Z}^d$, and a set of inliers for a given digital hyperplane. There exists an other hyperplane with the same inlier set such that it has at least d inliers as support points.

From *Prop.* 2, by taking all possible d-uplet of points in $\mathbf{S}$ as support points, one can construct the whole search space $\mathbb{F}(\mathbf{S})$ of $\mathbf{S}$ for digital hyperplane fitting.

Proposition 1. *Given* $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{Z}^d \mid i = 1..n\}$ *and a value* ω, *the number of digital hyperplanes* $\mathcal{DH}_\omega$ *generated from* $\mathbf{S}$ *is* $O(n^d)$.

Proof. From Def. 1 and *Prop.* 2, we need d linearly independent points as support points to determine a digital hyperplane $\mathcal{DH}_\omega$. In order to select d points in $\mathbf{S}$, we need a complexity of $O(n^d)$ since there are n points in $\mathbf{S}$ and the linearly independent test of these points is $O(1)$.

Let consider the two support hyperplanes of $\mathcal{DH}_\omega$:

$$\sum_{i=1}^{d} a_i x_i + a_{d+1} = 0 \tag{3}$$

$$\sum_{i=1}^{d} a_i x_i + a_{d+1} = \omega \tag{4}$$

It should be recalled that, for a given set of d support points, the digital hyperplane $\mathcal{DH}_\omega$ passing though them is not unique. Different cases may appear to the d selected support points of $\mathcal{DH}_\omega$. Typically, there are i points on the support hyperplane in Eq. 3, and $d-i$ points on the one in Eq. 4, for $i = 0, .., d$. In particular, $i = 0$ or $i = d$ mean all points belong respectively to Eq. 3 or Eq. 4. Due to the symmetry of selecting points, *e.g.* having i points on Eq. 3 and $d-i$ points on Eq. 4 is equivalent to $d-i$ points on Eq. 3 and i points on Eq. 4, the total number of possible hyperplanes $\mathcal{DH}_\omega$ passing though these d support points is $\binom{d}{0} + \frac{1}{2}\sum_{i=1}^{d-1}\binom{d}{i} + \binom{d}{d} = 1 + 2^{d-1}$. This leads to the final complexity of $O(2^{d-1}n^d)$ for generating all digital hyperplanes of a given set $\mathbf{S}$. For a fixed dimension space d, this complexity becomes $O(n^d)$. □

From *Propo.* 1, one can generate the whole search space $\mathbb{F}(\mathbf{S})$ from $\mathbf{S}$. Then, by verifying the inliers of each hyperplane in $\mathbb{F}(\mathbf{S})$, we can find the optimal hyperplane as the one that maximizes the number inliers. In this context, some solutions have been proposed for the specific cases of 2D and 3D, for instance in [34] a combinatorial approach using dual space is presented for digital line and plane fitting which a time complexity of $O(n^d \log n)$ for $d = 2, 3$, and an improved algorithm for 2D cases with $O(n^2)$ time-complexity using a topological sweep method in [20]. In [1], a study for efficient digital hyperplane fitting in $\mathbb{Z}^d$ with bounded error is investigated. Still in [1], a conjecture of optimal computational complexity for this problem in any dimension is provided, and it is $O(n^d)$.

4 Hyperplane fitting using dD Delaunay triangulation

The combinatorial approach in the previous section, by generating all possible digital hyperplanes from $\mathbf{S}$, allows to find the optimal solution for digital hyperplane fitting. It has, however, a high complexity of $O(n^d)$ with n is the number of discrete points in $\mathbf{S}$ and d is the dimension space of points in $\mathbf{S}$. This forbids the use of the method in many applications with big datasets.

Faced with this dilemma, a new approach of digital hyperplane fitting is proposed in this section. The approach is based on a heuristic involving the dD Delaunay triangulation. Roughly speaking, the method uses the triangulated elements to filter the *admissible* combinations of discrete points and to generate digital hyperplanes for the considered fitting problem.

One of the interesting aspect of the Delaunay triangulation is that it implicitly presents an information of distribution / density of points in the space; *i.e.* the points being close to each other form small and thin cells, while those being far create excessively large and long cells (see Fig. 4 for an illustration in 2D). This enables us to relate and to recognize points belonging to the same hyperplane; *i.e.* points appearing to lie reasonably on a hyperplane are close and arranged in a linear form. Roughly speaking, the fitting probblem can be solved using the dD Delaunay triangulation, according to two criteria: (1) the candidate hyperplanes should be on d-faces whose *width* is smaller than ω and (2) the best fitted hyperplane is the one containing the most number of inliers.

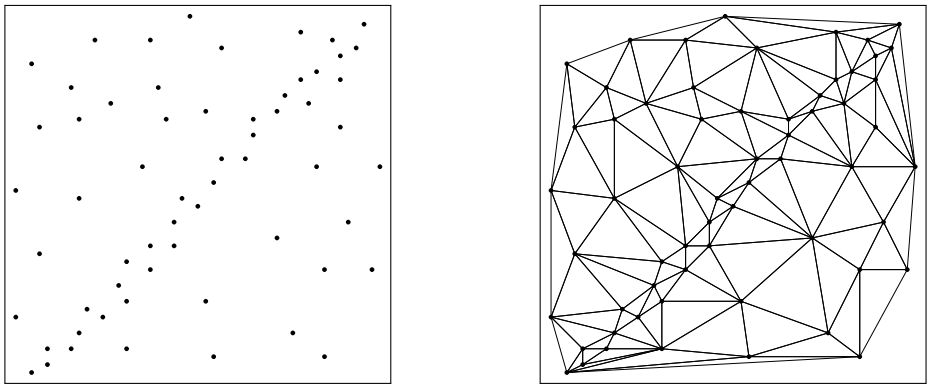


Fig. 4: A set of points (left) and its Delaunay triangulation (right). Points appearing to lie reasonably on a line are close and arranged in a linear form and their corresponding triangles by the Delaunay triangulation are smaller than the others.

Let $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{Z}^d \mid i = 1..n\}$ be the input set of points and $\mathcal{DT}(\mathbf{S})$ be the dD Delaunay triangulation of $\mathbf{S}$. Let t be a d-face in $\mathcal{DT}(\mathbf{S})$ such that t is formed by the vertices $\mathrm{y}_0, ..., \mathrm{y}_d \in \mathbf{S}$, w.l.o.g. t can be denoted by $t = (\mathrm{y}_0, ..., \mathrm{y}_d)$. We define the width –or height– of t as the minimal euclidean distance of a vertex of t to the hyperplane passing through all other vertices in t.

$$w(t) = \min \ \{||\mathrm{y}_j, \mathcal{H}(f_j)||_2^2 \text{ for } j = 0, ..., d\}$$

where $f_j = (\mathrm{y}_0, ..., \mathrm{y}_{j-1}, \mathrm{y}_{j+1}, ..., \mathrm{y}_d)$ is the hyperplane opposite to y_j in t. We call a d-face $t \in \mathcal{DT}(\mathbf{S})$ is *admissible* for the digital hyperplane fitting of $\mathbf{S}$ with a given ω if $w(t) \leq \omega$. This is called the *width condition* and will be used in the fitting method to filter the points and to generate the digital hyperplane for examination.

The main idea of the proposed method is as follows. We first compute the dD Delaunay triangulation $\mathcal{DT}(\mathbf{S})$ of $\mathbf{S}$, and then generate the digital hyperplanes from the admissible d-faces in $\mathcal{DT}(\mathbf{S})$; *i.e.* those that satisfy the width condition. For each computed digial hyperplane, we verify the inliers and report the optimal solution for the digital hyperplane fitting of $\mathbf{S}$ as the one maximizing the number of inliers. The algorithm is summarized in Algo. 1, and Fig. 5 illustrates the method in 2D (the idea is exactly the same in any dimension). It is stated in [26] the total number of faces in $\mathcal{DT}(\mathbf{S})$ is $O(n^{\lceil \frac{d}{2} \rceil})$. In other words, the d-faces in $\mathcal{DT}(\mathbf{S})$ being finite, this process has a guaranteed termination. Furthermore, the computations in Algo. 1 can be performed using only integer / rational numbers since all inputs are given in integer numbers.

By not generating and verifying all digital hyperplanes from $\mathbf{S}$ but only from the admissible d-faces in $\mathcal{DT}(\mathbf{S})$, this method leads to a much lower algorithmic complexity. More precisely, the worst-case complexity to compute the dD Delaunay triangulation of $\mathbf{S}$ is $O(n^{\lceil \frac{d}{2} \rceil + 1})$, finding admissible d-faces and generating the corresponding digital hyperplanes cost $O(n^{\lceil \frac{d}{2} \rceil})$, and the inliers verification

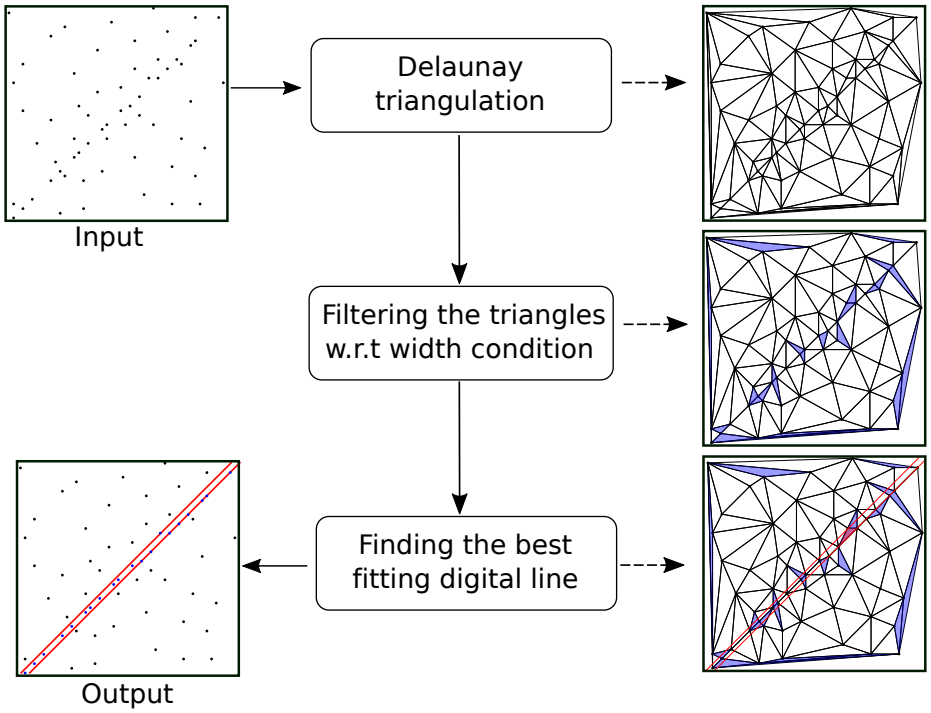


Fig. 5: Flowchart of the proposed method.

of each hyperplane is computed in $O(n)$. Therefore, the final computing complexity of Algo. 1 is $O(n^{\lceil \frac{d}{2} \rceil + 1})$ for n is the number of points in $\mathbf{S}$ and d is the dimension space of the points.

5 Experimental results

We have implemented in C++ the proposed method of fitting hyperplanes described in Sec. 4 using the Triangulations and Delaunay Triangulations package in CGAL [10]. This package provides functions to compute Delaunay triangulation of points in dimension 2, 3 and d. In particular, the dD Delaunay triangulation is computed by constructing convex hull in d+1 dimensions. This makes the method flexible and can handle any dimension. It is, however, much slower than the libraries specifically designed for 2D and 3D Delaunay Triangulation. As mentioned, the computation of dD Delaunay triangulation for n input points is at most $O(n^{\lceil \frac{d}{2} \rceil + 1})$, and it is $O(n)$ and $O(n^2)$ in 2D and 3D, respectively. In other words, the proposed algorithm is computationally more efficient in 2D and 3D with the specific implementations in CGAL. The source code is also available for testing at https://github.com/ngophuc/HyperplaneFitting.

In the following, we present the some experimental results in dimension 2, 3 and 4 to demonstrate the validity and efficiency of our method. It should be

Algorithm 1: Digital hyperplane fitting with fixed thickness

Algorithm `HyperplaneFitting`
- **Input**: A set $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{Z}^d \mid i = 1..n\}$ of n points and a value ω
- **Output**: The best fitted digital hyperplane
- **Variables** : $\mathcal{DT}(\mathbf{S})$: the set of d-faces of the Delaunay Triangulation of $\mathbf{S}$
  C: the set of d-faces satisfying width condition

2 $\mathcal{DT}(\mathbf{S}) = \{t_i = (\mathrm{y}_j)_{j=0}^{d}$ for $i = 1..m \mid y_j \in \mathbf{S}\}$

`/* Finding` d`-faces` t_i `satisfying the width condition */`

3 $C = \emptyset$

4 **foreach** $t_i \in \mathcal{DT}(\mathbf{S})$ **do**

5 **foreach** $\mathrm{y}_j \in t_i$ **do**

6 $f_j = (\mathrm{y}_0, ..., \mathrm{y}_{j-1}, \mathrm{y}_{j+1}, ..., \mathrm{y}_d)$ `//` $(d-1)$`-face opposite to vertex` y_j `in` t_i

7 $\mathcal{H}(f_j) = (a_i)_{i=0}^{d+1}$ `//the hyperplane passing through` d `points of` f_j `(see Eq. 1)`

8 $d = distance(\mathrm{y}_j, \mathcal{H}(f_j))$ `//the distance of` y_j `to` $\mathcal{H}(f_j)$

9 **if** $d^2 \leq \frac{w^2}{\sum_{i=1}^{d} a_i^2}$ **then**

10 $\mathcal{DH}_\omega(t_i) = (a_i)_{i=0}^{d+1}$ `//the digital hyperplane associated to` t_i `(see Eq. 2)`

11 $C = C \cup \mathcal{DH}_\omega(t_i)$

`/* Computing the best fitted digital hyperplane */`

12 $max = 0$

13 $\mathcal{DH}^*_\omega = (0, ..., 0)$

14 **foreach** $\mathcal{DH}^i_\omega \in C$ **do**

15 $n_i = \mathrm{CountInliers}(\mathbf{S}, \omega, \mathcal{DH}^i_\omega)$ `//(see the below Procedure)`

16 **if** $n_i > max$ **then**

17 $max = n_i$

18 $\mathcal{DH}^*_\omega = \mathcal{DH}^i_\omega$

19 return $\mathcal{DH}^*_\omega$

Procedure `CountInliers`
- **Input**: A set $\mathbf{S} = \{\mathrm{x}_i \in \mathbb{Z}^d \mid i = 1..n\}$ of n points, a value ω and a digital hyperplane $\mathcal{DH}_\omega = (a_i)_{i=0}^{d+1}$
- **Output**: The number of inliers of $\mathbf{S}$ w.r.t $\mathcal{DH}_\omega$

2 $count = 0$

3 **foreach** $\mathrm{x} = (x_0, x_1, ..., x_d) \in \mathbf{S}$ **do**

4 $v = \sum_{i=1}^{d} a_i x_i + a_{d+1}$

5 **if** $v \geq 0$ *and* $v \leq \omega$ **then**

6 $count = count + 1$

7 return *count*

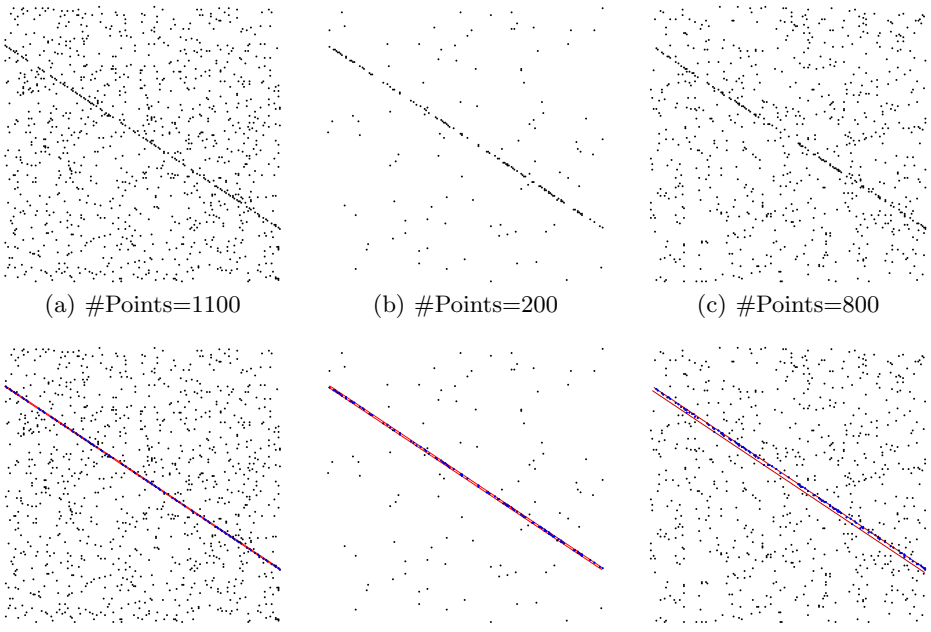
noticed that the proposed method is general and could work in any dimension, as well as its implementation remains conceptually unchanged in any dimension. Furthermore, the fitting problem in 3D and 4D is relatively expensive to solve as the runtime complexity is $O(n^3)$. All experiments are performed on a standard PC usinng Intel Core i5 processor.

5.1 2D case: Digital line fitting

At first, the experiments are carried out on 2D data points generated with the digital lines of equation

$$0 \leq 2x_1 + 3x_2 - 12 \leq \omega \tag{5}$$

with $\omega = 1, 2$ and 3. For each line, we randomly generate, according to Eq. 5, 100 inliers and $100k$ outliers with $k = 1, ..., 10$; *i.e.* 30 test datasets with ground-truth (see Fig. 6 for some examples). All data points are generated in a window of $[-100, 100]^2$. We report the inliers of the fitted line for each experiment and compare with the ground-truth by Eq. 5. Results are given in Tabs. 1 and 2.



(a) #Points=1100 (b) #Points=200 (c) #Points=800

(d) $\omega = 1$, #Inliers=100 (e) $\omega = 2$, #Inliers=103 (f) $\omega = 3$, #Inliers=108

Fig. 6: Evaluation on 2D synthetic data. First row: input points with 100 inliers, different numbers of outliers and different thickness ω. Second row: results of fitting digital lines obtained by the proposed method.

Figure	Thickness	#Points	#Inliers	Runtime
Fig. 6(a)	$\omega = 1$	1100	100	62.175 msec
Fig. 6(b)	$\omega = 2$	200	103	20.531 msec
Fig. 6(c)	$\omega = 3$	800	108	165.905 msec

Table 1: Results of fitted digital lines by the proposed method on Fig. 6.

Measures	Results
Runtime (in average)	71.69 msec
Precision (%): $P = \#(D \cap S)/\#D$	96.99 ± 4.1
Recall (%): $R = \#(D \cap S)/\#S$	100
F-measure (%): $F = 2 \times P \times R/(P + R)$	98.42 ± 2.2

Table 2: Measured performance of the proposed method on 2D synthetic data. S is the set of all ground-truth inliers, D the set of all inliers detected by the proposed method.

5.2 3D case: Digital plane fitting

Next experiments are on volume data points which are generated as follows. We consider digital planes of equation

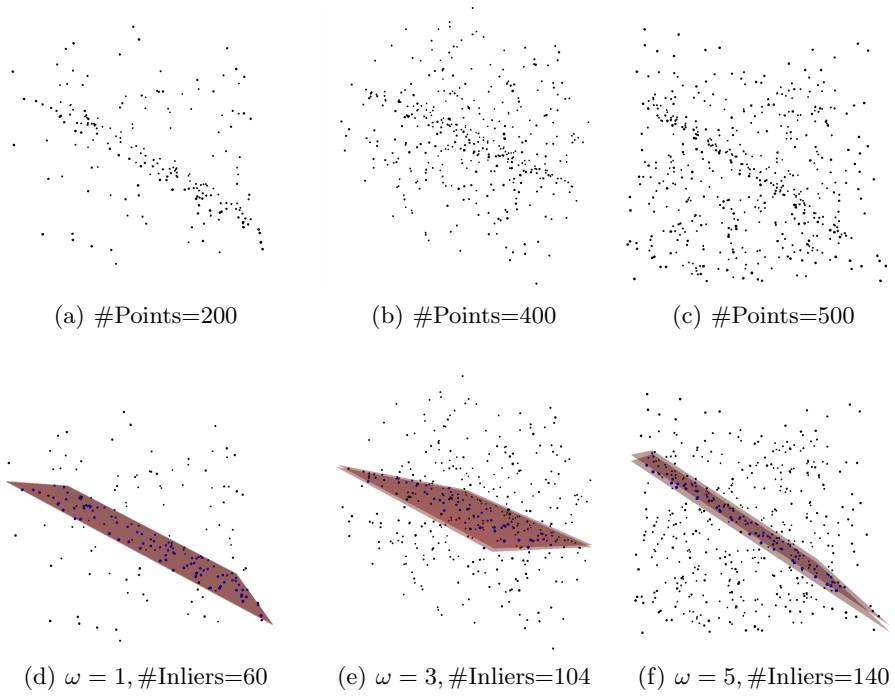
$$0 \leq 2x_1 + 3x_2 + x_3 - 9 \leq \omega \tag{6}$$

with $\omega = 1, 3$ and 5. Similarly to 2D, we randomly generate, for each plane, the 100 inliers and $100k$ outliers with $k = 1, ..., 10$. That makes 30 test datasets with ground-truth (see Fig. 7 for some examples). All data points are generated in a window of $[-100, 100]^3$. We report the inliers of the fitted planes and compare with the ground-truth by Eq. 6. Results are shown in Tabs. 3 and 4.

Furthermore, the proposed method allows to work with large datasets in an efficient way. As illustrated in Fig. 8 and Tab. 3, the algorithm takes around 23 seconds to deal with 2989 input points.

Figure	Thickness	#Points	#Inliers	Runtime
Fig. 7(a)	$\omega = 1$	200	60	35.291 msec
Fig. 7(b)	$\omega = 3$	400	104	172.838 msec
Fig. 7(c)	$\omega = 5$	800	140	1063.31 msec
Fig. 8(a)	$\omega = 1$	2989	578	21718.1 msec
Fig. 8(a)	$\omega = 3$	2989	1040	23328.6 msec

Table 3: Results of fitted digital planes by the proposed method on Fig. 7 and Fig. 8.



(a) #Points=200 (b) #Points=400 (c) #Points=500

(d) $\omega = 1, \#\text{Inliers}=60$ (e) $\omega = 3, \#\text{Inliers}=104$ (f) $\omega = 5, \#\text{Inliers}=140$

Fig. 7: Evaluation on 3D synthetic data. First row: input points with 100 inliers, different numbers of outliers and different thickness ω. Second row: results of fitting digital planes obtained by the proposed method.

Measures	Results
Runtime (in average)	533.68 msec
Precision (%): $P = \#(D \cap S)/\#D$	81.28 ± 15.92
Recall (%): $R = \#(D \cap S)/\#S$	77.7 ± 22.75
F-measure (%): $F = 2 \times P \times R/(P + R)$	76.16 ± 16.26

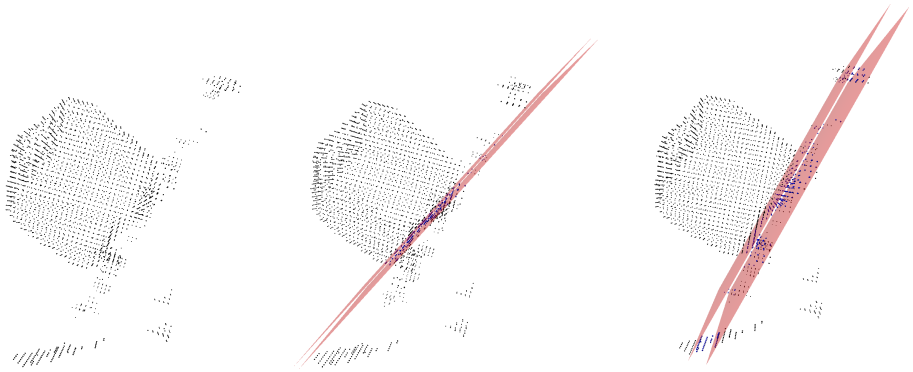
Table 4: Measured performance of the proposed method on 3D synthetic data. S is the set of all ground-truth inliers, D the set of all inliers detected by the proposed method.

5.3 4D case: Digital hyperplane fitting

Next experiments are on 4D data points generated with the following digital hyperplanes:

$$0 \leq 2x_1 + 3x_2 + x_3 + 7x_4 - 9 \leq \omega \tag{7}$$

with $\omega = 1, 3$ and 5. Then, we randomly generate, for each hyperplane, the 100 inliers and $100k$ outliers with $k = 1, ..., 10$. That makes 30 test datasets with ground-truth. All data points are generated in a window of $[-100, 100]^4$. We



(a) #Points=2989 (b) $\omega = 1, \#\text{Inliers}=578$ (c) $\omega = 3, \#\text{Inliers}=1040$

Fig. 8: Evaluation on 3D data: (a) input points, (b) and (c) are fitting digital planes of (a) obtained by the proposed method for $\omega = 1$ and 3, respectively.

Measures	Results
Runtime (in average)	11582.41 msec
Precision (%): $P = \#(D \cap S)/\#D$	73.84 ± 20.54
Recall (%): $R = \#(D \cap S)/\#S$	69.58 ± 31.3
F-measure (%): $F = 2 \times P \times R/(P + R)$	67.62 ± 24.04

Table 5: Measured performance of the proposed method on 4D synthetic data. S is the set of all the ground-truth inliers, D the set of all the detected inliers.

report the inliers of the fitted hyperplane for each experiment and compare with the ground-truth by Eq. 7. Results are shown in Tab. 5.

Overall, the experiments demonstrate the efficiency and effectiveness of the proposed method. It is robust to the number of outliers and has a good performance in term of runtime. In particular, the proposed method can be applied to large datasets and in high dimensions, which are difficult with traditional methods. However, the experiment was conducted mostly with synthetic data. This allows us to evaluate the behaviour of the proposed method. In practice, the sets of input points –particularly, in 2D and 3D– can be obtained by feature extraction or segmentation algorithm.

6 Conclusion

This paper presented methods of digital hyperplane fitting in $\mathbb{Z}^d$ of given thickness ω. Two strategies have been proposed. The first one consists of generating all possible digital hyperplanes from a set $\mathbf{S}$ of n points. Then, performing an exhaustive search overall generated hyperplanes allows to find the global op-

timum of the fitting problem. However, this approach costs $O(n^d)$ which is a polynomial complexity of degree equal to the dimension of the problem. This limits its use in practical contexts. To overcome this issue, we proposed another method with a heuristic based on Delaunay triangulation to find a *local optimum* of digital hyperplane fitting problem. More precisely, instead of examining all digital hyperplanes generated from $\mathbf{S}$, we verify only the hyperplanes generated from the d-cells of the Delaunay triangulation of $\mathbf{S}$ whose width is smaller than ω. This method leads to a much lower algorithmic complexity of $O(n^{\lceil \frac{d}{2} \rceil + 1})$ and it is efficient in dealing with large datasets. Furthermore, the presented method can be applied to points in $\mathbb{R}^d$ with no special change.

Experiments have been conducted to validate the feasibility of the proposed method. Nonetheless, it is mostly with synthetic data. In future works, we would like to test the proposed method on real data and to provide comparisons with other methods in the literature such as [1, 16, 18, 34]. Another perspective is the application of the proposed method for shape fitting problem. As it is shown in [25], by a transformation of the model formulation, the digital plane fitting can be used to solve digital annulus fitting.

References

1. Aiger, D., Kenmochi, Y., Talbot, H., Buzer, L.: Efficient robust digital hyperplane fitting with bounded error. In: Proceedings of DGCI. vol. LNCS 6607, pp. 223–234 (2011)
2. Amenta, N., Attali, D., Devillers, O.: A tight bound for the delaunay triangulation of points on a polyhedron. Discrete & Computational Geometry 48, 19–38 (2012)
3. Amenta, N., Choi, S., Dey, T.K., Leekha, N.: A simple algorithm for homeomorphic surface reconstruction. In: International Journal of Computational Geometry and Applications. pp. 213–222 (2000)
4. Arlinghaus, S.L.: Practical Handbook of Curve Fitting. CRC Press (1994)
5. Barber, C.B., Dobkin, D.P., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. 22(4), 469–483 (1996)
6. Bern, M., Eppstein, D.: Mesh generation and optimal triangulation. In: Lecture Notes Series on Computing, Computing in Euclidean Geometry. pp. 47–123 (1995)
7. Boissonnat, J.D., Devillers, O., Hornus, S.: Incremental construction of the delaunay graph in medium dimension. In: Annual Symposium on Computational Geometry. pp. 208–216 (2009)
8. Boissonnat, J.D., Yvinec, M.: Algorithmic geometry (1998), cambridge University Press,UK
9. Cevikalp, H.: Best fitting hyperplanes for classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(6), 1076–1088 (2016)
10. CGAL-Team: Cgal: The computational geometry algorithms library (2019), http://www.cgal.org
11. Chernov, N.: Circular and linear regression: Fitting circles and lines by least squares. CRC Pres (2010)
12. Chum, O.: Two-view geometry estimation by random sample and consensus (2005), phD thesis, Czech Technical University, Prague, Czech Republic
13. Delaunay, B.: Sur la sphère vide (1934), bulletin de l'Acadmie des Sciences de l'URSS, Classe des Sciences Mathmatiques et Naturelles

14. Dey, T.K.: Curve and surface reconstruction : Algorithms with mathematical analysis. Cambridge University Press, New York (2006)
15. Duda, R., Hart, P.: Use of the hough transformation to detect lines and curves in pictures. Communications of the ACM 15, 11–15 (1972)
16. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)
17. Gajic, Z.: Prentice Hall (2003)
18. Hough, P.V.C.: Machine analysis of bubble chamber pictures. In: International. Conference on High Energy Accelerators and Instrumentation. pp. 554–556 (1959)
19. Kenmochi, Y., Buzer, L., Sugimoto, A., Shimizu, I.: Discrete plane segmentation and estimation from a point cloud using local geometric patterns. Journal of Automation and Computing 5(3), 246–256 (2008)
20. Kenmochi, Y., Talbot, H., Buzer, L.: Efficiently computing optimal consensus of digital line fitting. In: Proceedings of ICPR. pp. 1064–1067 (2010)
21. Köster, K., Spann, M.: An approach to robust clustering - application to range image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(2), 430–444 (2000)
22. Musin, O.R.: Properties of the delaunay triangulation. In: Proceedings of the thirteenth annual symposium on Computational geometry (1997)
23. Ngo, P., Nasser, H., Debled-Rennesson, I.: A discrete approach for decomposing noisy digital contours into arcs and segments. In: Proceedings of RRPR, in workshop of ACCV. vol. LNCS 10117, pp. 493–505 (2016)
24. NIST/SEMATECH: e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/ (2012)
25. Phan, M.S., Kenmochi, Y., Sugimoto, A., Talbot, H., Andres, E., Zrour, R.: Efficient robust digital annulus fitting with bounded error. In: Proceedings of DGCI. vol. LNCS 7749, pp. 253–264 (2013)
26. Raimund, S.: The upper bound theorem for polytopes: an easy proof of its asymptotic version. Computational Geometry 5, 115–116 (1995)
27. Rajan, V.T.: Optimality of the delaunay triangulation in r^d. Discrete and Computational Geometry 12(1), 189–202 (1994)
28. Reveillès, J.P.: Géométrie discrète, calculs en nombre entiers et algorithmique (1991), thèse d'état. Université Louis Pasteur, Strasbourg
29. Rousseeuw, P.J.: Least median of squares regression. Journal of the American statistical association 79(388), 871–880 (1984)
30. Searle, S.R., Gruber, M.H.: Linear Models, 2nd Edition. Wiley (2016)
31. Shum, H., Ikeuchi, K., Reddy, R.: Principal component analysis with missing data and its application to polyhedral object modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(9), 854–867 (1995)
32. Sivignon, I., Dupont, F., Chassery, J.M.: Decomposition of a three-dimensional discrete object surface into discrete plane pieces. Algorithmica 38(1), 25–43 (2004)
33. Zitová, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing 21(11), 977–1000 (2003)
34. Zrour, R., Kenmochi, Y., Talbot, H., Buzer, L., Hamam, Y., Shimizu, I., Sugimoto, A.: Optimal consensus set for digital line and plane fitting. International Journal of Imaging Systems and Technology 21(1), 45–57 (2011)